ЛАБОРАТОРНА РОБОТА № 1

Взаємодія між процесами.

Розподіл даних між процесами.

Робота з файлами, які відображуються у пам'ять

Мета: отримання базових навичок з взаємодії між процесами, розподілу даних між процесами. Навчитися працювати з файлами які відображуються у пам'ять.

Хід роботи:

Завдання №1: Необхідно написати дві програми (три), які будуть мати спільні дані та одночасно до них звертатися.

Існує кілька механізмів реалізації спільного доступу до даних різних процесів. Скористаємося одним з них, найбільш зручним - проектуванням файлу в пам'ять. Одна програма буде сортувати дані у файлі, а інша відображати вміст цього файлу. Працювати обидва процеси будуть одночасно. Третя програма буде створювати (або заповнювати по-новому) масив випадкових чисел.

Створіть файл data.dat. У ньому мають бути записані числа, згенеровані випадковим чином. Кількість чисел - 20-30 штук. Діапазон значень: від 10 до 100. (Це саме числа, а не символьні рядки зберігають ASCII коди цифр !!!)

Лістинг програми (додаток 1):

```
using System;
using System.IO;
using System.IO.MemoryMappedFiles;
using System.Text;
using System.Threading;

namespace Program3
```

```
{
    class Program
    {
        static byte size = 30;
        static void Main(string[] args)
        {
            Console.OutputEncoding = Encoding.Unicode;
            Console.InputEncoding = Encoding.Unicode;
            try
            {
                MemoryMappedFile mmf = MemoryMappedFile.OpenExisting("Numbers");
                Semaphore semaphore = Semaphore.OpenExisting("NumbSem");
                Console.WriteLine("Натисніть пробіл для сортування(не закривайте додаток
Program2, що відкрився)");
                while (Console.ReadKey().Key != ConsoleKey.Spacebar)
                {
                    Console.WriteLine("Помилка! Натисніть, будь ласка, пробіл");
                }
                Console.Clear();
                Console.WriteLine("Зачекайте, йде сортування...");

                var stream = mmf.CreateViewStream();
                var handle = stream.SafeMemoryMappedViewHandle;
                unsafe
                {
                    byte* pointer = null;
                    handle.AcquirePointer(ref pointer);

                    for (int i = 1; i < size; i++)
                    {
                        for (int j = 0; j < size - i; j++)
                        {
                            try
                            {
                                semaphore.WaitOne();

                                if (*(pointer + j) < *(pointer + j + 1))
                                {
                                    Swap(ref *(pointer + j), ref *(pointer + j + 1));
                                }
                            }
                            finally
                            {
                                semaphore.Release();
                            }
                            Thread.Sleep(100);
                        }
                    }
                }
                Console.WriteLine("Робота завершена(додаток 3)");
                Console.ReadLine();
            }
            catch (FileNotFoundException)
            {
                Console.WriteLine("Помилка, Program1 не запущено");
                Console.ReadLine();
            }
            catch (WaitHandleCannotBeOpenedException)
            {
                Console.WriteLine("Semaphore не створено, запустіть Program1");
                Console.ReadLine();
            }

        }
```

```
        static void Swap(ref byte a, ref byte b)
        {
            var t = a;
            a = b;
            b = t;
        }
    }
       }
```

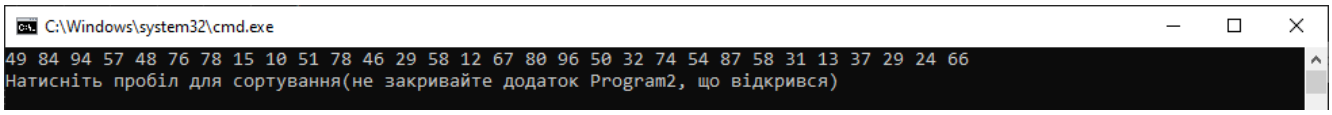

Рис. 1. Результат виконання програми (додаток 1)

Лістинг програми (додаток 2):

```
using System;
using System.IO.MemoryMappedFiles;
using System.Threading;
using System.Windows.Forms;

namespace Program2
{
    public partial class Form1 : Form
    {
        static byte size = 30;
        MemoryMappedFile memory = MemoryMappedFile.OpenExisting("Numbers");
        Semaphore semaphore = Semaphore.OpenExisting("NumbSem");

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            Render();
            timer1.Enabled = true;
            timer1.Start();
        }
        public void Render()
        {

            try
            {
                semaphore.WaitOne();

                string res = "";
                var stream = memory.CreateViewStream();
                var handle = memory.CreateViewStream().SafeMemoryMappedViewHandle;
                unsafe
                {
                    byte* pointer = null;
                    handle.AcquirePointer(ref pointer);

                    for (int i = 0; i < size; i++)
                    {
                        for (int j = 0; j < *(pointer + i); j++)
                            res += "*";
```

```
                        res += $" {*(pointer + i)}";
                        res += "\n";
                    }
                    label1.Text = res;
                }
            }
            finally
            {
                semaphore.Release();
            }
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            Render();
        }
    }
}
```

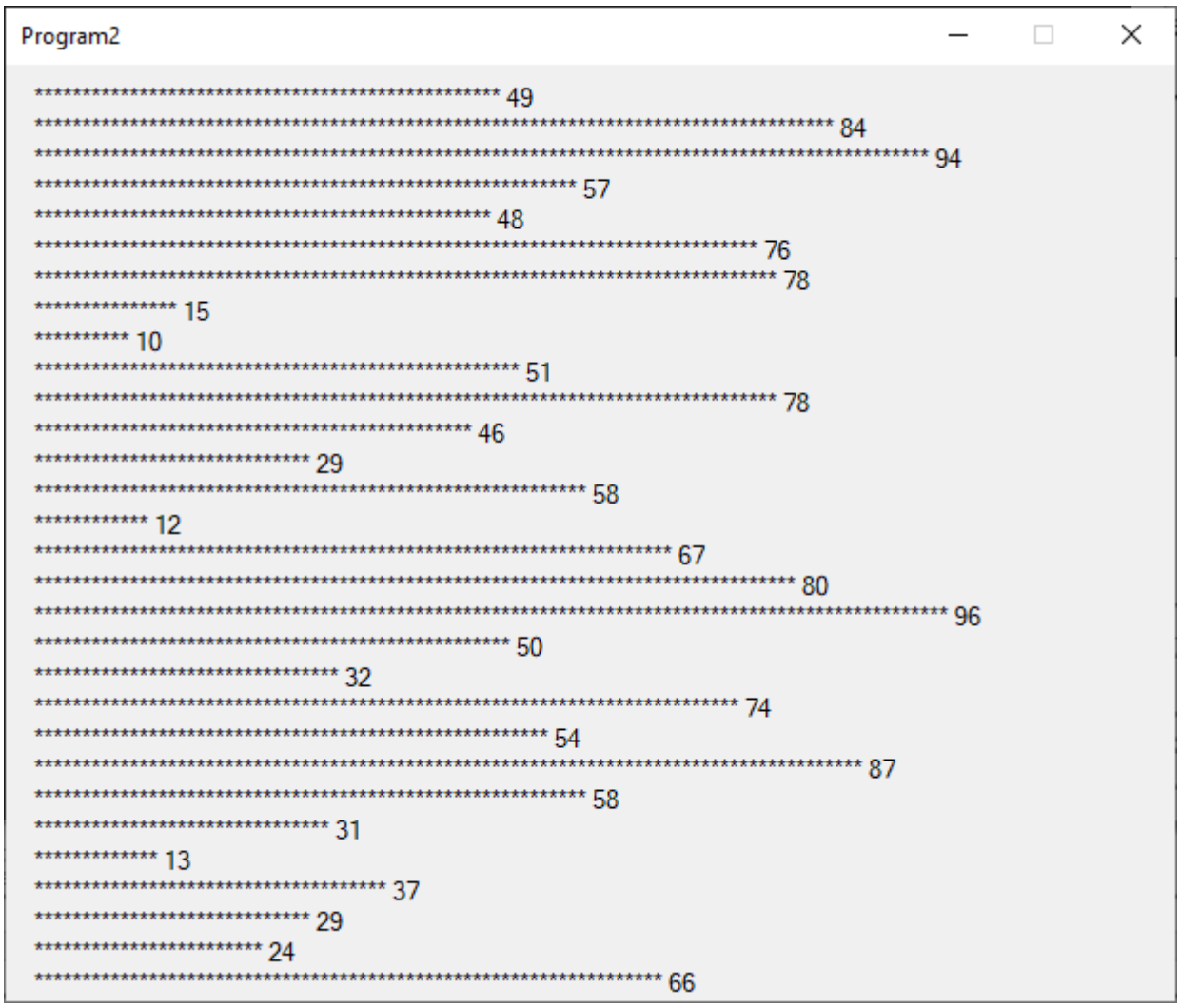

Рис. 2. Результат виконання програми (додаток 2)

Завдання №2: Для коректної роботи зі спільними даними у цих двох програмах потрібно додати синхронізацію потоків, які можуть одночасно звертатися до спільних даних.

Для організації такої синхронізації потрібно використати об'єкт ядра ОС mutex або semaphor, або інший синхронізуючий об'єкт, а також функції очікування (наприклад, WaitForSinglJbject()).

Також обов'язковим є використання обробки виняткових ситуацій в роботі вище описаних трьох програм. Бо, некоректна робота будь якої з трьох, викличе неправильну роботу інших, через блокування спільних даних.

Для обробки виняткових ситуацій, необхідно правильно визначити критичні секції коду усіх написаних програм.

Лістинг програми (додаток 3):

```
using System;
using System.IO;
using System.IO.MemoryMappedFiles;
using System.Text;
using System.Threading;

namespace Program3
{
    class Program
    {
        static byte size = 30;
        static void Main(string[] args)
        {
            Console.OutputEncoding = Encoding.Unicode;
            Console.InputEncoding = Encoding.Unicode;
            try
            {
                MemoryMappedFile mmf = MemoryMappedFile.OpenExisting("Numbers");
                Semaphore semaphore = Semaphore.OpenExisting("NumbSem");
                Console.WriteLine("Натисніть пробіл для сортування(не закривайте додаток
Program2, що відкрився)");
                while (Console.ReadKey().Key != ConsoleKey.Spacebar)
                {
                    Console.WriteLine("Помилка! Натисніть, будь ласка, пробіл");
                }
                Console.Clear();
                Console.WriteLine("Зачекайте, йде сортування...");

                var stream = mmf.CreateViewStream();
                var handle = stream.SafeMemoryMappedViewHandle;
                unsafe
                {
                    byte* pointer = null;
                    handle.AcquirePointer(ref pointer);

                    for (int i = 1; i < size; i++)
                    {
                        for (int j = 0; j < size - i; j++)
```

```
                        {
                            try
                            {
                                semaphore.WaitOne();

                                if (*(pointer + j) < *(pointer + j + 1))
                                {
                                    Swap(ref *(pointer + j), ref *(pointer + j + 1));
                                }
                            }
                            finally
                            {
                                semaphore.Release();
                            }
                            Thread.Sleep(100);
                        }
                    }
                }
                Console.WriteLine("Робота завершена(додаток 3)");
                Console.ReadLine();
            }
            catch (FileNotFoundException)
            {
                Console.WriteLine("Помилка, Program1 не запущено");
                Console.ReadLine();
            }
            catch (WaitHandleCannotBeOpenedException)
            {
                Console.WriteLine("Semaphore не створено, запустіть Program1");
                Console.ReadLine();
            }

        }
        static void Swap(ref byte a, ref byte b)
        {
            var t = a;
            a = b;
            b = t;
        }
    }
   }
```

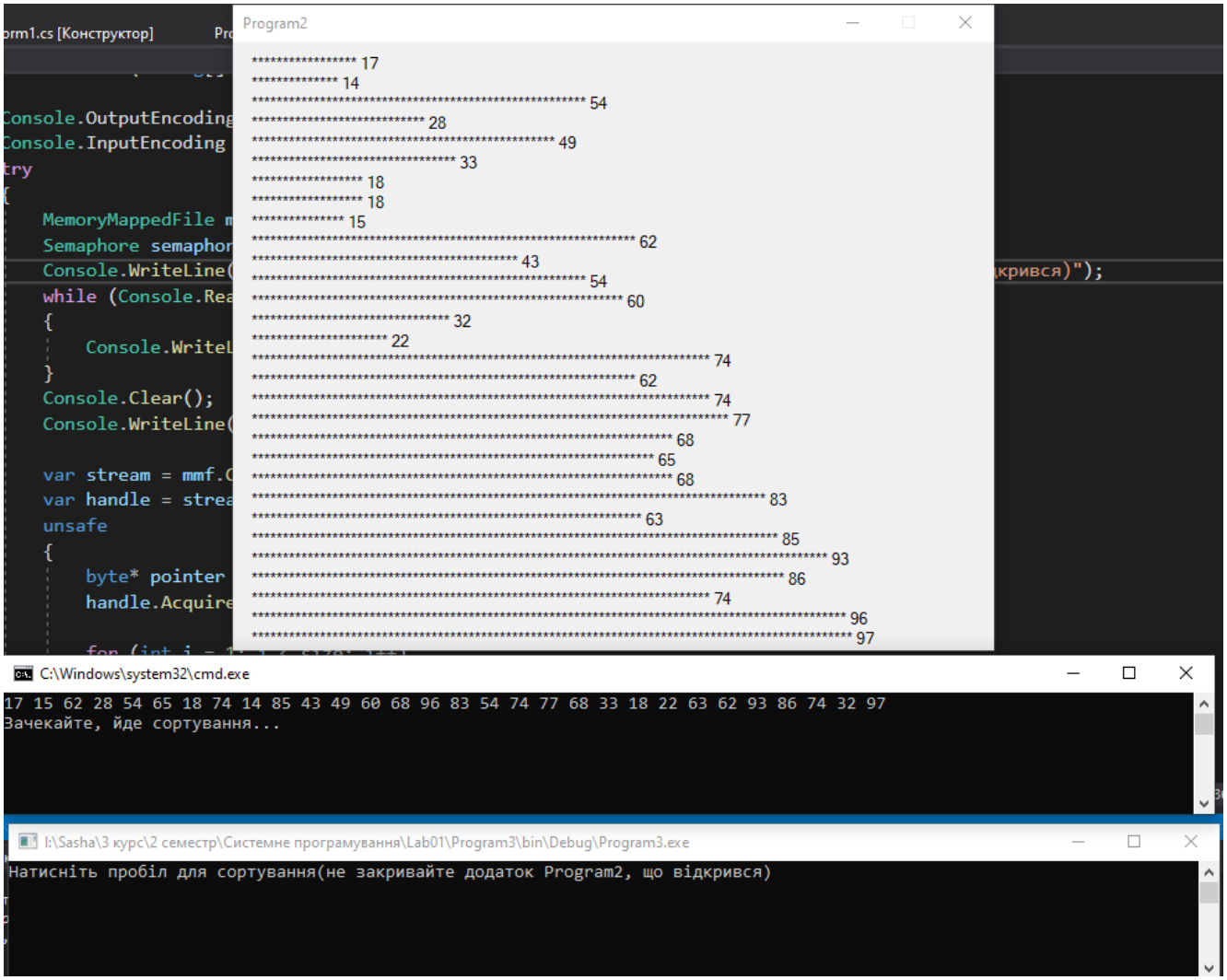

Рис. 3. Результат виконання програм

Висновки: було отримано базові навички з взаємодії між процесами, розподілу даних між процесами. Було вивчено основи роботи з файлами, які відображуються у пам'ять.